

Случайные процессы. Прикладной поток.

Практическое задание 1

Правила:

- Выполненную работу нужно отправить на почту `probability.diht@yandex.ru`, указав тему письма "[СП17] Фамилия Имя - Задание 1". Квадратные скобки обязательны. Вместо Фамилия Имя нужно подставить свои фамилию и имя.
- Прислать нужно ноутбук и его pdf-версию. Названия файлов должны быть такими: `1.N.ipynb` и `1.N.pdf`, где N - ваш номер из таблицы с оценками.
- Никакой код из данного задания при проверке запускаться не будет.
- Дедлайн и система оценивания будут объявлены позже.



В Британской империи в Викторианскую эпоху (1837—1901) было обращено внимание на вымирание аристократических фамилий. В связи с этим в своей статье в *The Educational Times* в 1873 году Гальтон поставил вопрос о вероятности вымирания фамилии. Решение этого вопроса нашел Ватсон и вместе в 1874 году они написали статью "On the probability of the extinction of families". На сайте wikitree.com (<http://wikitree.com>) в свободно распространяемом формате собрано большое количество данных о родословных различных людей. В коллекции есть как люди, жившие во времена поздней античности, так и наши современники. На основе некоторой части этих данных вам предстоит провести исследование о вымирании фамилий.

Вам предоставляются несколько файлов, в которых содержатся данные о некоторых родословных. Вам предстоит проводить исследование на нескольких из этих файлов (каких именно, см. в таблице). Формат файлов следующий:

```
generation \t name \t gender \t birthday \t deathdate \t parents \t siblings \t spouses \t children
```

Эти данные означают номер поколения, фамилию, пол, дату рождения, дату смерти, родителей, братьев и сестер, супруг, детей соответственно. Если какая-то характеристика неизвестна (кроме номера поколения и фамилии), вместо нее ставится пустая подстрока. Если каких-то характеристик несколько, то они разделены через ";". Все люди представлены некоторым идентификатором <id>, который соответствует адресу <http://www.wikitree.com/wiki/<id>>. Например, идентификатор Романов - 29 соответствует адресу <http://www.wikitree.com/wiki/Romanov-29> (<http://www.wikitree.com/wiki/Romanov-29>). В файле родословные отделяются друг от друга пустой строкой.

Для облегчения вашей работы мы предоставляем вам код, который считывает данные из этого файла и преобразует их в список ветвящихся процессов. Каждый ветвящийся процесс содержит список списков, в каждом из которых содержатся все люди из соответствующего поколения. Обратите внимание, что одни и те же родословные могут попасть в разные файлы. В таком случае их можно считать разными, но при желании вы можете удалить копии.

В предоставленных данных в каждой родословной для каждого мужчины на следующем поколении содержатся все его дети, которые были указаны на сайте. Для женщин дети в данной родословной не указаны. Это связано с тем, что женщины обычно меняют свою фамилию, когда выходят замуж, тем самым, они переходят в другую ветку. С точки зрения ветвящихся процессов, нужно иметь в виду, что если у мужчины родилось 3 мальчика и 4 девочки, то у него 3 потомка как продолжателя фамилии.

Ваша задача --- исследовать процесс вымирания фамилий на основе предложенных данных. В данном задании вам предстоит сделать оценку закона размножения, а в следующем задании --- провести остальной анализ.

```
In [16]: import numpy as np
import scipy.stats as sps
from collections import Counter # это может пригодиться
from BranchingProcess import Person, BranchingProcess, read_from_files

import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams.update({'font.size': 16})
%matplotlib inline
```

1. Описательный анализ

Большая часть кода, необходимая для проведения данного анализа, является технической и основывается на работе с пакетом BranchingProcess. Поэтому данный код полностью вам выдается, вам нужно только выполнить его, подставить имена файлов. Кроме того, код анализа позволит вам лучше понять структуру данных.

Считайте данные с помощью предложенного кода. Посчитайте количество родословных.

```
In [269]: with open("Варианты.txt", "r") as f:
          for s in f.readlines():
              if ("Шевкунов" in s):
                  print(s)
```

```
594 Шевкунов Кирилл Сергеевич АД      ==-      P O N I C R T J S E
```

```
In [23]: first_letters = "P O N I C R T J S E"
          fn_prefix = "./data/"
          fn_postfix = ".txt"
          file_names = []
          for c in first_letters:
              if c != " ":
                  file_names.append(fn_prefix + c + fn_postfix)

          for v in file_names:
              print(v)
```

```
./data/P.txt
./data/O.txt
./data/N.txt
./data/I.txt
./data/C.txt
./data/R.txt
./data/T.txt
./data/J.txt
./data/S.txt
./data/E.txt
```

```
In [24]: processes = read_from_files(file_names)
          print(len(processes))
```

```
55337
```

В имеющихся данных очень много людей, про которых известно лишь то, что они когда-то существовали. Обычно их фамилия неизвестна (вместо фамилии у них может стоять, к примеру, В-290), а у некоторых из них неизвестен даже пол, не говоря уже о родителях и детях. Такие данные стоит удалить.

Удалите все процессы, состоящие только из одного поколения (в котором, естественно, будет только один человек). Сколько осталось процессов?

```
In [25]: for i in range(len(processes))[::-1]:  
        if len(processes[i].generations) < 2:  
            del processes[i]  
  
print(len(processes))
```

15841

Для лучшего понимания задачи и предложенных данных посчитайте следующие характеристики: минимальное, максимальное и среднее число поколений в роду, год рождения самого старого и самого молодого человека, среднюю продолжительность жизни.

```
In [26]: # Пробный вывод для лучшего понимания происходящего
# str(processes[0])
for generation in processes[0].generations:
    print("__GENERATION__")
    for man in generation:
        print(man)
```

____GENERATION____						
Murdoch-533	male	1740--	1822-Feb-18		Rodger-269;Ross-7636	Murdoch-527;Murdoch-534;Murdoch-535;Murdoch-536;Murdoch-537;Murdoch-538
____GENERATION____						
Murdoch-527	male	1792-Mar-28	1852-Jun-05	Murdoch-533;Rodger-269	Murdoch-534;Murdoch-535;Murdoch-536;Murdoch-537;Murdoch-538 Richardson-9305 Murdoch-515;Murdoch-528	
Murdoch-534	male	1800-Mar-08		Murdoch-533;Ross-7636	Murdoch-527;Murdoch-535;Murdoch-536;Murdoch-537;Murdoch-538	
Murdoch-535	female	1804-Feb-11		Murdoch-533;Ross-7636	Murdoch-527;Murdoch-534;Murdoch-536;Murdoch-537;Murdoch-538	
Murdoch-536	female	1806-Jan-26	1863--	Murdoch-533;Ross-7636	Murdoch-527;Murdoch-534;Murdoch-535;Murdoch-537;Murdoch-538 Jamieson-955 Jamieson-956;Jamieson-957;Jamieson-958;Jamieson-959;Jamieson-960;Jamieson-961;Jamieson-962	
Murdoch-537	female	1807-Dec-13		Murdoch-533;Ross-7636	Murdoch-527;Murdoch-534;Murdoch-535;Murdoch-536;Murdoch-538	
Murdoch-538	male	1809-Oct-12		Murdoch-533;Ross-7636	Murdoch-527;Murdoch-534;Murdoch-535;Murdoch-536;Murdoch-537	
____GENERATION____						
Murdoch-515	male	1816-Feb-09	1864-Jul-24	Murdoch-527;McWhinnie-15	Murdoch-528	McConnell-1855
Murdoch-528	male	1828--	1873-Jun-17	Murdoch-527;Richardson-9305	Murdoch-515	
____GENERATION____						
Murdoch-529	male	1849--	1925-Dec-26	Murdoch-515;McConnell-1855	Murdoch-530;Murdoch-531;Murdoch-532;Murdoch-513 Wallace-7710 Murdoch-678	
Murdoch-530	male	1852--	1878-Aug-07	Murdoch-515;McConnell-1855	Murdoch-529;Murdoch-531;Murdoch-532;Murdoch-513	
Murdoch-531	male	1854--	1885-Jun-22	Murdoch-515;McConnell-1855	Murdoch-529;Murdoch-530;Murdoch-532;Murdoch-513 Wilson-35755;Wilson-34382 Murdoch-740;Murdoch-742;Murdoch-743;Murdoch-716	
Murdoch-532	male	1856-Jan-28	1869-Sep-28	Murdoch-515;McConnell-1855	Murdoch-529;Murdoch-530;Murdoch-531;Murdoch-513	
Murdoch-513	male	1858-Dec-09	1931-Dec-13	Murdoch-515;McConnell-1855	Murdoch-529;Murdoch-530;Murdoch-531;Murdoch-532 Hutchinson-2956 Murdoch-514;M-1000	
____GENERATION____						
Murdoch-678	male	1884--	1959-Apr-26	Murdoch-529;Wallace-7710	Fletcher-5489	Murdoch-679
Murdoch-740	male	1879--		Murdoch-531;Wilson-34382	Murdoch-742;Murdoch-743;Murdoch-716	
Murdoch-742	female	1879--		Murdoch-531;Wilson-34382	Murdoch-740;Murdoch-743;Murdoch-716	
Murdoch-743	male	1880--	1945--	Murdoch-531;Wilson-34382	Murdoch-740;Murdoch-742;Murdoch-716	
Murdoch-716	female	1886-Jan-07	1947-Sep-22	Murdoch-531;Wilson-34382	Murdoch-740;Murdoch-716	

742;Murdoch-743	Fraser-3629	Fraser-2380			
Murdoch-514	female 1899--	1932-Jul-01	Murdoch-513;Hutchinson-2956	M-1000	
M-1000	male		Murdoch-513;Hutchinson-2956	Murdoch-514	P-445
<u> GENERATION </u>					
Murdoch-679	female	1916-Oct-19	2013-Oct-09	Murdoch-678;Fletcher-5489	Harvey-6695
P-445	female		M-1000		

```
In [27]: generation_counts = []
years = []

for pedigree in processes:
    generation_counts.append(len(pedigree.generations))

    for generation in pedigree.generations:
        for person in generation:
            if person.birthday != '':
                years.append(person.birthday.split('-')[0])

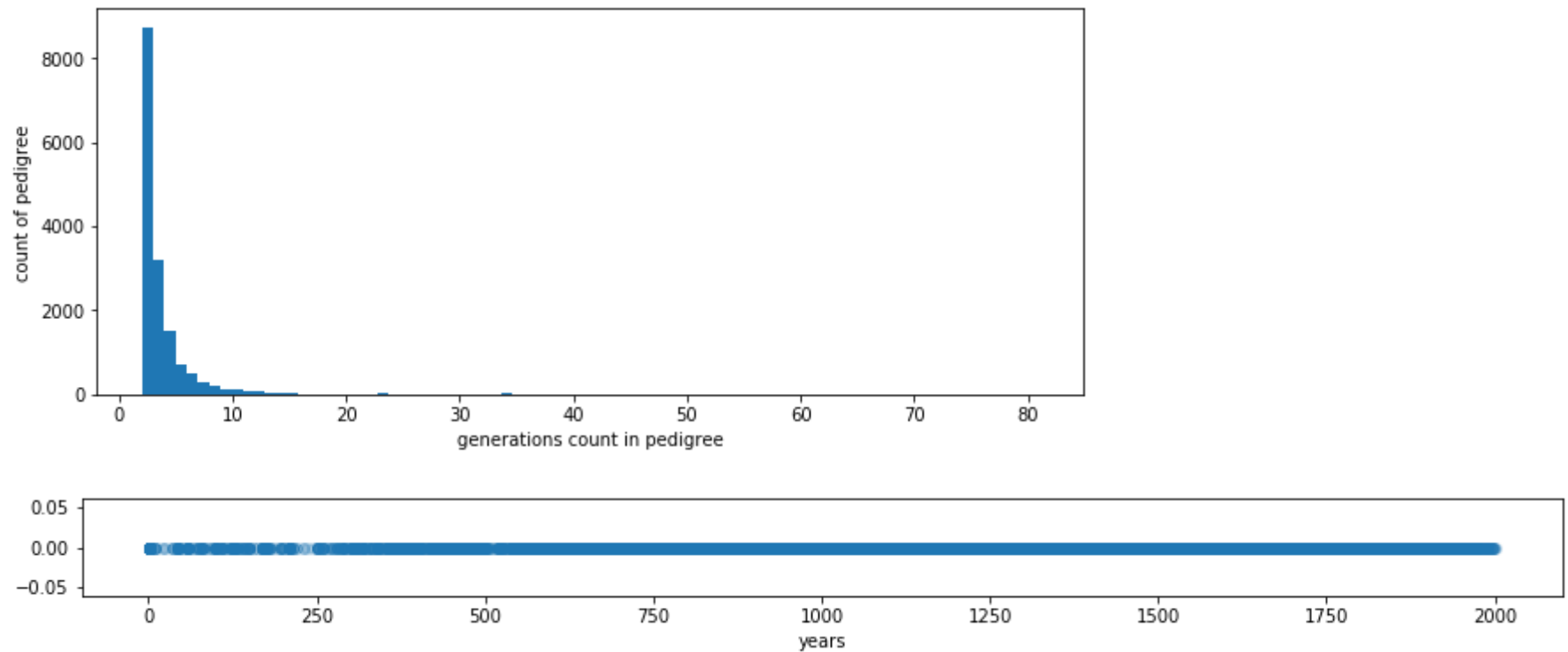
years = np.array(years, dtype=int)
print('Минимальное число поколений в роду:', min(generation_counts))
print('Максимальное число поколений в роду:', max(generation_counts))
print('Среднее число поколений в роду:', round(np.mean(generation_counts), 1))
print('Год рождения самого старого:', min(years))
print('Год рождения самого молодого:', max(years))
```

```
Минимальное число поколений в роду: 2
Максимальное число поколений в роду: 81
Среднее число поколений в роду: 3.4
Год рождения самого старого: 1
Год рождения самого молодого: 2000
```

Постройте гистограмму зависимости количества поколений в родословной от количества родословных. На следующем графике отложите на временной оси года рождения всех людей.

```
In [28]: plt.figure(figsize=(10, 4))
plt.hist(generation_counts, bins=80)
plt.xlabel('generations count in pedigree')
plt.ylabel('count of pedigree') # было pedigree
plt.show()

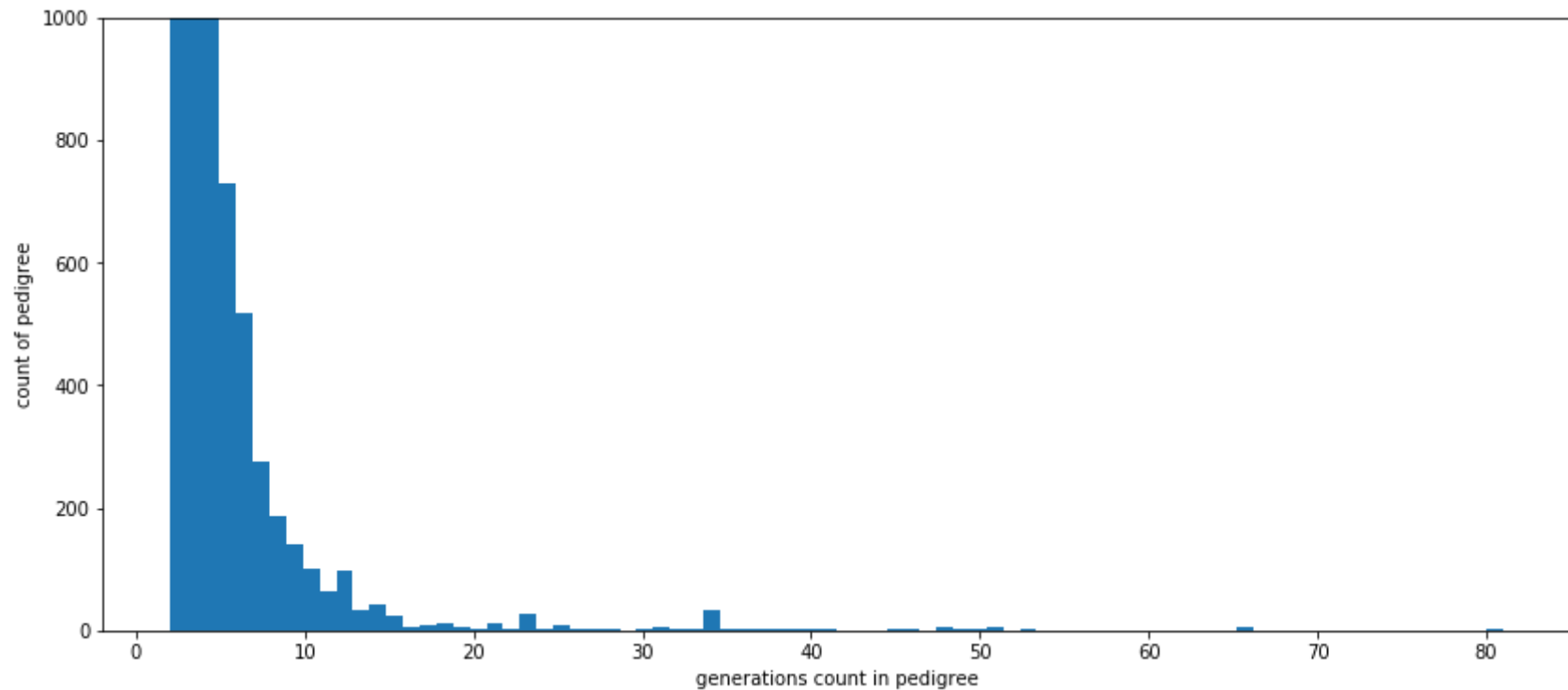
plt.figure(figsize=(15, 1))
plt.scatter(years, np.zeros_like(years), alpha=0.2)
plt.xlabel('years')
plt.show()
```

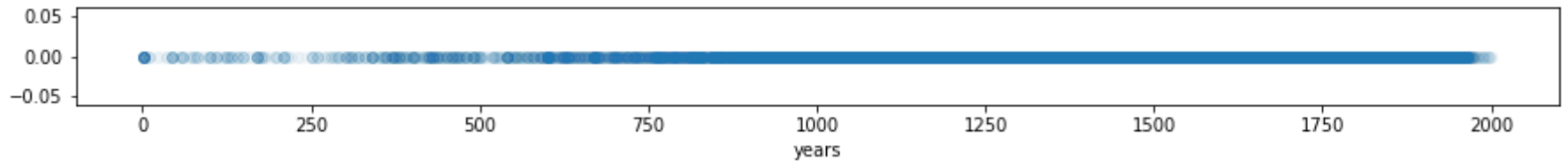



```
In [29]: plt.figure(figsize=(14, 6)) # was figsize=(10,4)
plt.ylim(ymax=1000)
plt.hist(generation_counts, bins=80)
plt.xlabel('generations count in pedigree')
plt.ylabel('count of pedigree') # было pedogree

plt.show()

plt.figure(figsize=(15, 1))
plt.scatter(years, np.zeros_like(years), alpha=0.025) # was alpha=0.2
plt.xlabel('years')
plt.show()
```





Посчитайте среднюю продолжительность жизни.

```
In [30]: ages = []
for pedigree in processes:
    for generation in pedigree.generations:
        for person in generation:
            if person.birthday != '' and person.deathdate != '':
                ages.append(int(person.deathdate.split('-')[0]) - \
                           int(person.birthday.split('-')[0]))

### ! В коде выше мы берём разность года рождения и года смерти,
### ! т.е. продолжительности жизни оценена с ошибкой до года,
### ! хотя среднее должно пострадать не сильно

mean_age = np.mean(ages)
print(round(mean_age, 2))
```

56.56

2. Оценка закона размножения

Для начала предположим, что все выданные вам процессы являются частью одного большого процесса с общим предком. В следующем задании рассмотрим так же случай, когда все процессы являются разными.

Чтобы проводить какой-либо анализ ветвящегося процесса нужно некоторым образом оценить закон размножения. Кажется, что для этого достаточно посчитать количество сыновей у каждого человека, получив тем самым выборку неотрицательных целых чисел. Однако, проблема в том, что данные неполные, в частности, некоторые поля могут быть не заполнены. Тем не менее обычно у человека указаны либо все дети, либо не указаны вообще. Таким образом, условно мы можем разделить выборку на две части: поле детей заполнено (в т.ч. если у человека на самом деле нет детей), поле детей незаполнено. Если бы первая часть выборки была бы полностью известна, что распределение можно оценить по ней. Нам же неизвестен размер выборки и количество нулевых элементов в ней. Количество положительных элементов известно.

Математическая постановка задачи

P_θ --- неизвестное распределение из некоторого класса распределений \mathcal{P} на \mathbb{Z}_+ .

X_1, \dots, X_n --- выборка из распределения P_θ , причем n и количество нулей в выборке неизвестны.

Y_1, \dots, Y_s --- положительная подвыборка, которая полностью нам известна. В нашей задаче Y_j --- количество сыновей у j -го человека среди тех, у кого есть хотя бы один сын.

Оценку параметра θ можно найти методом максимального правдоподобия:

$$\prod_{i=1}^s P_\theta(Y_i | Y_i > 0) \rightarrow \max_{\theta}$$

В качестве классов распределений \mathcal{P} рассмотрите пуассоновское и геометрическое распределения. По желанию можете рассмотреть другие классы распределений, осмысленные в данной задаче

Внимание! Применение метода `fit` из `scipy.stats` является некорректным в данной задаче, поскольку рассматривается усеченная выборка. Задачу максимизации нужно решить явно, выписав все формулы (которые тоже нужно прислать вместе с кодом).

После оценки параметров проведите проверку принадлежности неизвестного распределения рассматриваемому семейству распределений \mathcal{P} с помощью критерия хи-квадрат, взяв для него то распределение из \mathcal{P} , которое соответствует оценке максимального правдоподобия. Постарайтесь учесть все особенности проверки гипотез, которые обсуждались на семинаре. Для каждого класса постройте также график частот и функции $P_\theta(y | Y > 0)$.

Оценка ММП

1. Геометрическое распределение

$P_p(X = n) = (1 - p)^n p$ (Геометрическое распределение с нулём, $X \in \{0, 1, 2, \dots\}$)

Для положительных Y_i имеем: $P_p(Y_i|Y_i > 0) = \frac{P(Y_i)}{P(Y_i > 0)} = \frac{P(Y_i)}{1 - P(Y_i = 0)} = \frac{(1-p)^n p}{1-p} = (1-p)^{n-1} p$, т.е. положительные Y_i распределены по геометрическому распределению без нуля.

$$\prod_{i=1}^s P_p(Y_i|Y_i > 0) = \prod_{i=1}^s (1-p)^{Y_i-1} p = (1-p)^{\sum_{i=1}^s Y_i - s} p^s$$

$$\Rightarrow \left(\sum_{i=1}^s Y_i - s \right) \ln(1-p) + s \ln p$$

Возьмём производную для нахождения точки максимума (из вида логарифмической функции правдоподобия ясно, что при $p \in [0, 1]$ и положительных Y_i (коэффициенты при логарифмах неотрицательны) наблюдается один максимум):

$$\left(\sum_{i=1}^s Y_i - s \right) \frac{-1}{1-p} + \frac{s}{p} = 0$$

$$\frac{s}{1-p} - \frac{\sum_{i=1}^s Y_i}{1-p} + \frac{s}{p} = 0$$

$$\frac{sp + s - sp}{(1-p)p} = \frac{\sum_{i=1}^s Y_i}{1-p}$$

$$p = \frac{s}{\sum_{i=1}^s Y_i} = \frac{1}{\bar{Y}}$$

Итого: $p^* = 1/\bar{Y}$ - оценка по ММП для геометрического распределения

2. Пуассоновское распределение

$$P_\lambda(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}; P_\lambda(X = 0) = e^{-\lambda}; P_\lambda(X > 0) = 1 - e^{-\lambda}$$

$$\prod_{i=1}^s P_{\lambda}(Y_i | Y_i > 0) = \prod_{i=1}^s \frac{\frac{\lambda^{Y_i}}{Y_i!} e^{-\lambda}}{1 - e^{-\lambda}} = \frac{\frac{\sum_{i=1}^s Y_i}{\prod_{i=1}^s Y_i!} e^{-\lambda n}}{(1 - e^{-\lambda})^n}$$

$$\Rightarrow L(\lambda, Y) = \sum_{i=1}^s Y_i \ln \lambda - n\lambda - \ln\left(\prod_{i=1}^s Y_i!\right) - n \ln(1 - e^{-\lambda})$$

$$\Rightarrow L'(\lambda, Y) = \sum_{i=1}^s Y_i \frac{1}{\lambda} - n - n \frac{e^{-\lambda}}{(1 - e^{-\lambda})} = \sum_{i=1}^s Y_i \frac{1}{\lambda} - \frac{n}{(1 - e^{-\lambda})} =: 0$$

Оценка по ММП достигается в корне:

$$\bar{Y} = \frac{\lambda}{(1 - e^{-\lambda})}$$

Получена возрастающая на $\lambda \in (0, +\infty)$ функция ($\frac{d}{dx} \frac{x}{(1-e^{-x})} = \frac{e^x(-x+e^x-1)}{(e^x-1)^2} > 0$), при этом $\bar{Y} \geq s \geq 1$, а $\lim_{\lambda \rightarrow +0} (\frac{\lambda}{(1-e^{-\lambda})}) = 1$ (по Тейлору), значит у этого уравнения есть корень, при том один. Его можно найти двоичным поиском или поиском по сетке.

Итого: Оценка по ММП достигается в корне (он выражается только через страшные функции и его предлагается искать численными методами, благо функция хорошо для этого подходит):

$$\bar{Y} = \frac{\lambda}{(1 - e^{-\lambda})}$$

Проверка теоретических выкладок практическим методом (самостоятельность)

Для проверки предлагается взять числа из некоторых геометрического и пуассоновского распределения и вычислить оценку ММП по сгенерированному данным, сравнить её с исходным параметром "на глаз" (разумно было бы провести проверку множество раз с различными параметрами и размерами выборок, проверяя стат. значимость, например, тем же тестом хи-квадрат, но этот раздел не требуется, его просто жалко удалять)

```
In [14]: import scipy.stats as sps
import scipy.optimize
import numpy as np # слишком долго листать на верх
```

```
In [13]: X = sps.poisson(mu=777.).rvs(size=1000)
Y = np.array([x for x in X if x > 0])

print(X[:10])
print(Y[:10])

def f(x):
    return x / (1. - np.exp( - x)) - Y.mean()

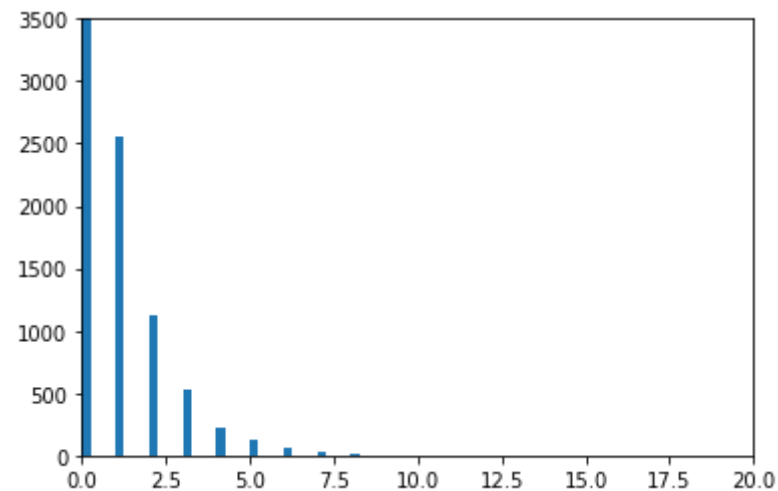
scipy.optimize.root(f,2.) # найденный x должен ~= mu

[762 737 780 776 797 784 759 809 749 813]
[762 737 780 776 797 784 759 809 749 813]
```

```
Out[13]: fjac: array([[ -1.]])
fun: array([ 0.])
message: 'The solution converged.'
nfev: 5
qtf: array([ 175.125])
r: array([ -1.])
status: 1
success: True
x: array([ 777.125])
```

```
In [19]: p = 0.1 + np.random.rand() * 0.8 # ~ U[0.1, 0.9]
print("p = ", p)
X = sps.geom(p=p).rvs(size=10000) - 1
Y = np.array([x for x in X if x > 0])
plt.ylim(ymax=3500)
plt.xlim(xmax=20)
plt.hist(X, bins = 40)
plt.show()
print("X[:15] =", X[:15])
print("Y[:10] =", Y[:10])
```

p = 0.5374137464214082



```
X[:15] = [0 0 0 0 0 0 0 4 0 0 2 3 1 3 1]
Y[:10] = [4 2 3 1 3 1 1 2 3 2]
```

```
In [21]: theta = 1. / np.array(Y).mean()
print("p = ", p)
print("theta = ", theta) # должны быть приблизительно равны
```

p = 0.5374137464214082
theta = 0.544880785414

Обработка данных

Составим, для начала Y

```
In [42]: male = set()

Y_names = list()
for pedigree in processes:
    for generation in pedigree.generations:
        for person in generation:
            if person.gender == "male":
                male.add(person.name)

male
```



```
In [43]: Y = list()

for pedigree in processes:
    for generation in pedigree.generations:
        for person in generation:
            if person.gender == "male":
                child_cnt = 0
                for child in person.children:
                    if child in male:
                        child_cnt += 1
                if (child_cnt > 0):
                    Y.append(child_cnt)

print(Y[:10])

[3, 2, 5, 1, 2, 1, 1, 1, 1, 2]
```

```
In [44]: print("len(Y)", len(Y))
Y_data_mean = np.array(Y).mean()

len(Y) 49266
```

Найдём оценку ММП для пуассоновского распределения:

```
In [45]: def poiss_tf(x):  
        assert type(Y) == list # вдруг  
        return x / (1. - np.exp(-x)) - Y_data_mean  
result = scipy.optimize.root(poiss_tf, 20.)  
result
```

```
Out[45]: fjac: array([[ -1.]])  
        fun: array([ 8.88178420e-16])  
        message: 'The solution converged.'  
        nfev: 8  
        qtf: array([ -1.01263575e-09])  
        r: array([ -0.77214533])  
        status: 1  
        success: True  
        x: array([ 1.80957111])
```

```
In [46]: mu_data = result.x[0]  
mu_data
```

```
Out[46]: 1.8095711098638814
```

Найдём оценку ММП для геометрического распределения:

```
In [47]: p_data = 1. / Y_data_mean  
p_data
```

```
Out[47]: 0.46214025740122316
```

```
In [49]: from scipy.stats import chisquare  
print(len(Y))
```

```
49266
```

Применим тест хи-квадрат для полученных данных. Как обсуждалось на семинаре, большой размер выборки нежелателен, в силу ряда причин. Предложено было использовать выборки по сто элементов, что мы и сделаем.

```
In [56]: from math import factorial
```

```
In [153]: sub_Y = np.array(np.array(Y)[sps.randint.rvs(0, len(Y), size=100)])

def to_freq_table(sub_Y, ddof=1):
    Y_table = np.zeros(np.array(sub_Y).max() + 1)
    for y in sub_Y:
        Y_table[y] += 1

    YY = []
    for i in range(ddof, len(Y_table)):
        if (Y_table[i] < 5):
            YY = Y_table[ddof:i]
            YY[-1] += Y_table[i:].sum()
            break
    return YY

YY = to_freq_table(sub_Y)

print("Количество отцов с (i+1) детьми, с укрупнёнными группами = ", YY)

# p_data = 1. / sub_Y.mean()
```

```
Количество отцов с (i+1) детьми, с укрупнёнными группами = [ 59.  17.  11.   5.   8.]
```

```

In [154]: geom = []
          pois = []

def geom_cond_pmf(x,p):
    q = 1. - p
    return q ** (x - 1) * p

def pois_cond_pmf(x,mu):
    e = np.exp(- mu)
    return mu ** x * e / (1. - e) / (factorial(x))

for i in range(1, len(Y)):
    geom.append(geom_cond_pmf(x=i, p=p_data))
    pois.append(pois_cond_pmf(x=i, mu=mu_data))

geom.append(1. - np.array(geom).sum())
pois.append(1. - np.array(pois).sum())
geom = np.array(geom) * Y.sum()
pois = np.array(pois) * Y.sum()
print("Y (данные выборки для теста) = ", Y)
print("geom (теоретическая выборка из геометрического) = ", geom)
print("pois (теоретическая выборка из пуассоновского) = ", pois)

print("\nТест для геометрического:\n", sps.chisquare(Y, geom))
print("\nТест для пуассоновского:\n", sps.chisquare(Y, pois))

Y (данные выборки для теста) = [ 59.  17.  11.   5.   8.]
geom (теоретическая выборка из геометрического) = [ 50.25125628  24.9993687   12.43687187   6.18718751   6.
12531564]
pois (теоретическая выборка из пуассоновского) = [ 35.42741384  32.05421229  19.33479217   8.74692033   4.4
3666137]

Тест для геометрического:
Power_divergenceResult(statistic=5.0503757481607652, pvalue=0.28216757650103302)

Тест для пуассоновского:
Power_divergenceResult(statistic=30.814777114070829, pvalue=3.3396090728281857e-06)

```

```
In [131]: # вычисленная руками для самопроверки статистика  
          ((YY - geom) ** 2 / geom).sum()
```

```
Out[131]: 1.5022630518015581
```

Итого: при уровне значимости $\alpha = 0.05$ (он определён заранее, хотя впервые фигурирует здесь)

- в первом тесте нулевая гипотеза, состоящая в том, что законом размножения является геометрическое распределение с оценённым выше параметром (на самом деле, некоторое его упрощение), не может быть отвергнута ($pvalue > \alpha$)
- во втором тесте нулевая гипотеза, состоящая в том, что законом размножения является пуассоновское распределение с оценённым выше параметром (на самом деле, некоторое его упрощение), должно быть отвергнуто ($pvalue < \alpha$)

Графики:

```

In [150]: P_geom = [geom_cond_pmf(x,p_data) for x in range(1, 11)]
P_pois = [pois_cond_pmf(x,mu_data) for x in range(1, 11)]

plt.figure(figsize=(14,7))

plt.subplot(221)
plt.hist(range(10), weights=P_geom, ls='solid', ec='black')
plt.title("$\\mathsf{P}_\\theta$ (y \\left| Y > 0 \\right)$ для геометрического")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))

plt.subplot(222)
plt.hist(range(10), weights=P_pois, ls='solid', ec='black')
plt.title("$\\mathsf{P}_\\theta$ (y \\left| Y > 0 \\right)$ для пуассоновского")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))

def top_ten(X):
    X = list(X)
    while (len(X) < 10):
        X.append(0)
    return np.array(X[:10])

plt.show()
plt.figure(figsize=(14,7))

plt.subplot(221)
FT = top_ten(to_freq_table(Y))
FT = FT / FT.sum() * 100.
plt.hist(range(10), weights=FT, ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n " +
        "на 100 человек, среди тех, у кого они есть) по всей выборке")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))

plt.show()
plt.figure(figsize=(14,7))

plt.subplot(221)
FT = [sps.geom(p=p_data).pmf(i+1)*100 for i in range(0, 10)]
plt.hist(range(10), weights=FT, ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n на" +

```

```

        " 100 человек, среди ВСЕХ) при геометрическом\n" +
        "распределении из теории и оценки ММП")
plt.xticks(np.linspace(0.5, 8.5, 10), range(0, 10))

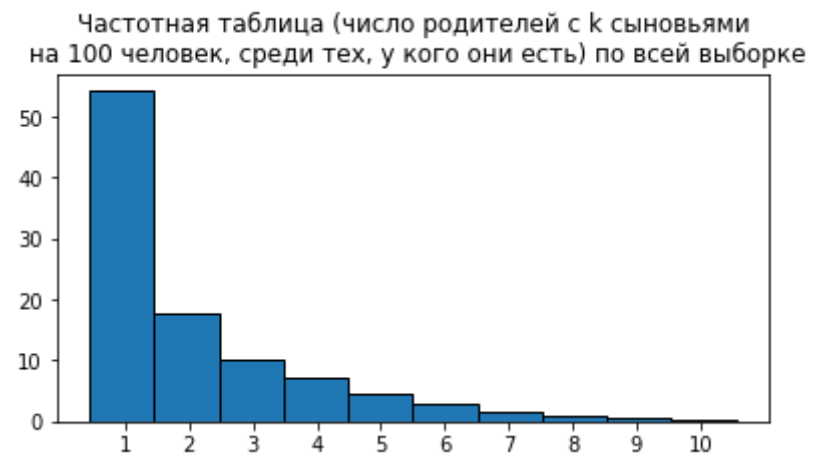
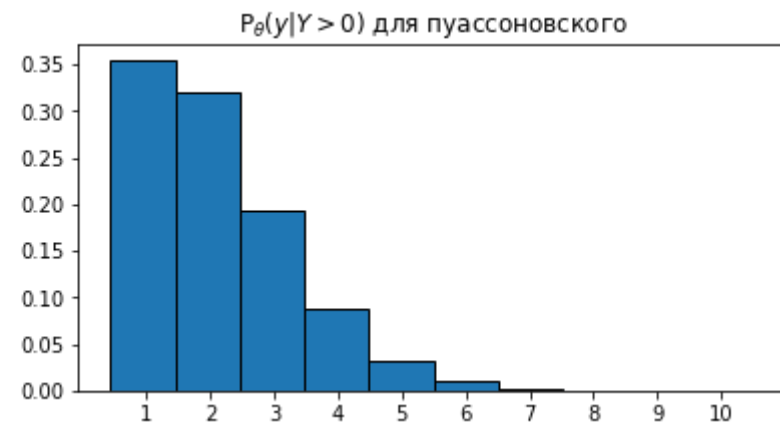
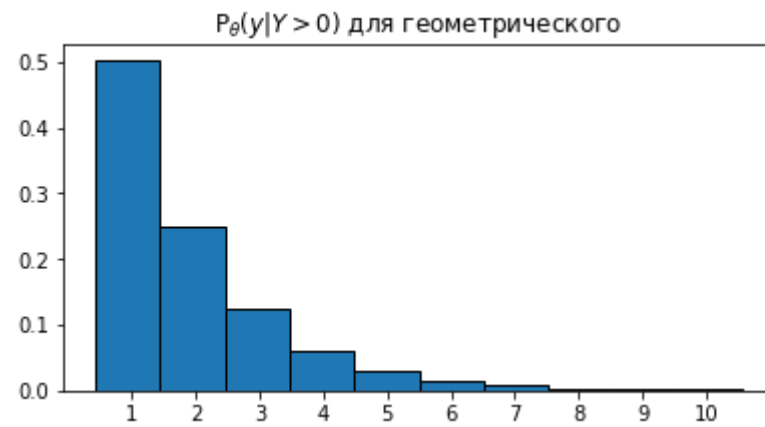
plt.subplot(222)
FT = [sps.poisson(mu=mu_data).pmf(i)*100 for i in range(0, 10)]
plt.hist(range(10), weights=FT, ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n" +
        "на 100 человек, " +
        "среди ВСЕХ) при пуассоновском\n" +
        "распределении из теории и оценки ММП")
plt.xticks(np.linspace(0.5, 8.5, 10), range(0, 10))

plt.show()

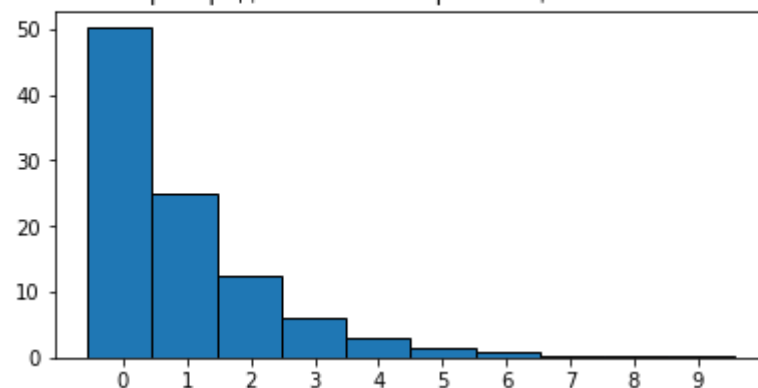
print("Данные для теста хи-квадрат:")

plt.figure(figsize=(14,7))
plt.subplot(221)
plt.hist(range(10), weights=top_ten(YY), ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n" +
        "на 100 человек, среди тех, у кого они есть) по сокращённой\n" +
        "выборке (последняя группа включает все последующие)")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))
plt.subplot(222)
plt.hist(range(10), weights=top_ten(geom), ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n" +
        "на 100 человек, среди тех, у кого они есть) по теоретическому\n" +
        "геометрическому распределению\n(последняя группа включает все последующие)")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))
plt.show()
plt.figure(figsize=(14,7))
plt.subplot(221)
plt.hist(range(10), weights=top_ten(pois), ls='solid', ec='black')
plt.title("Частотная таблица (число родителей с k сыновьями\n" +
        "на 100 человек, среди тех, у кого они есть) по теоретическому\n" +
        "пуассоновскому распределению\n(последняя группа включает все последующие)")
plt.xticks(np.linspace(0.5, 8.5, 10), range(1, 11))
plt.show()

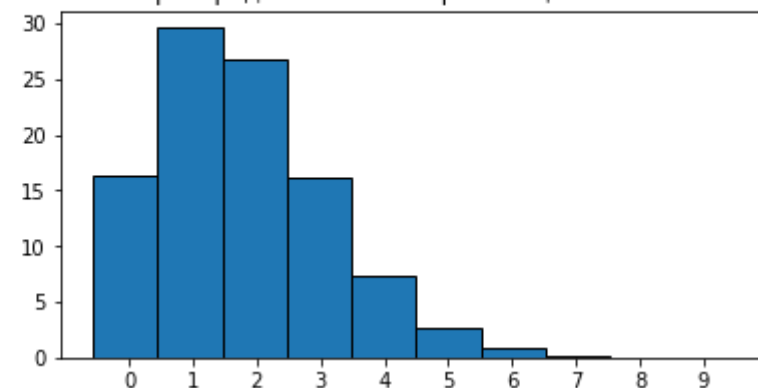
```



Частотная таблица (число родителей с k сыновьями на 100 человек, среди ВСЕХ) при геометрическом распределении из теории и оценки ММП

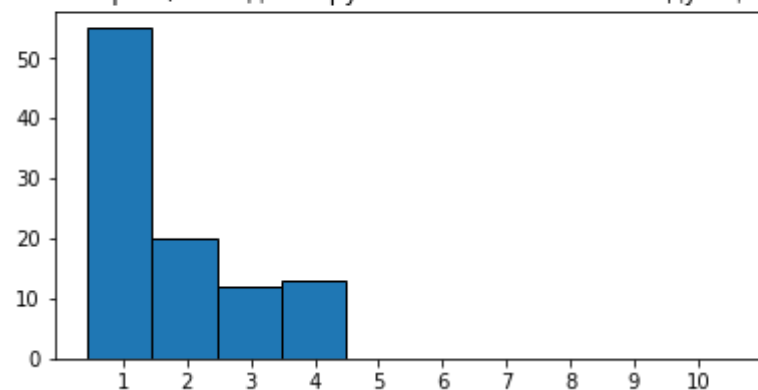


Частотная таблица (число родителей с k сыновьями на 100 человек, среди ВСЕХ) при пуассоновском распределении из теории и оценки ММП

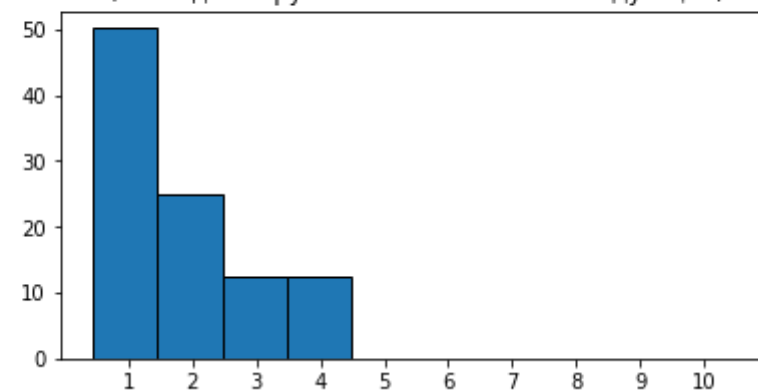


Данные для теста хи-квадрат:

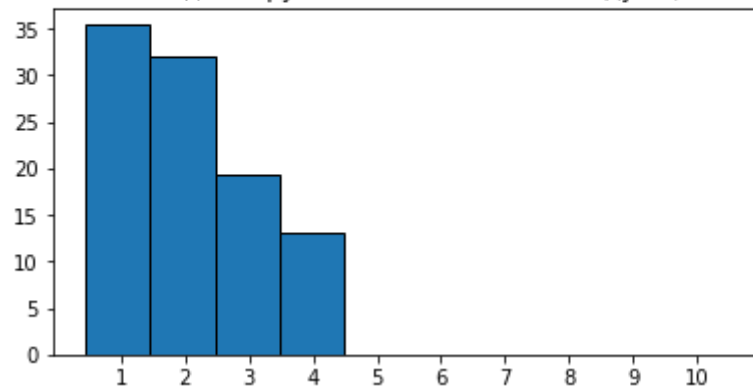
Частотная таблица (число родителей с k сыновьями на 100 человек, среди тех, у кого они есть) по сокращённой выборке (последняя группа включает все последующие)



Частотная таблица (число родителей с k сыновьями на 100 человек, среди тех, у кого они есть) по теоретическому геометрическому распределению (последняя группа включает все последующие)



Частотная таблица (число родителей с k сыновьями
на 100 человек, среди тех, у кого они есть) по теоретическому
пуассоновскому распределению
(последняя группа включает все последующие)



Комментарий к графикам: видно, что геометрическое распределение приближает лучше, и что условная вероятность подсчитана приблизительно корректно (можно вырезать участок из графика с нулями в детях и растянуть их на относительное изменение числа человек в сумме - получится график для условной вероятности)

Построим графики $P_{\theta}(Y|Y > 0)$, численно считая, что это функция от действительного, а не натурального (с нулём) числа, отмечая значения при целочисленном аргументе точками. (ещё один график для красоты, чтоб был)

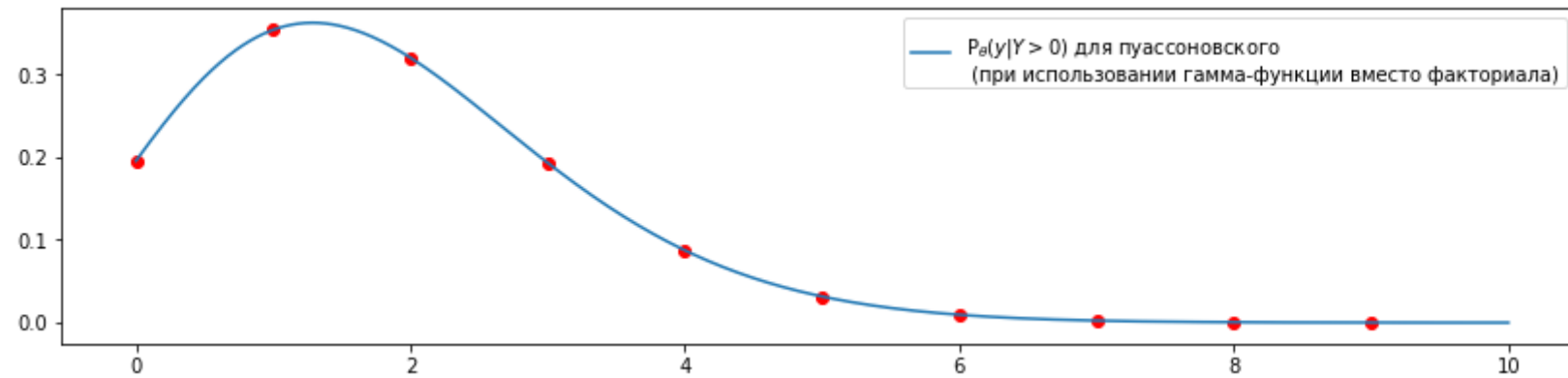
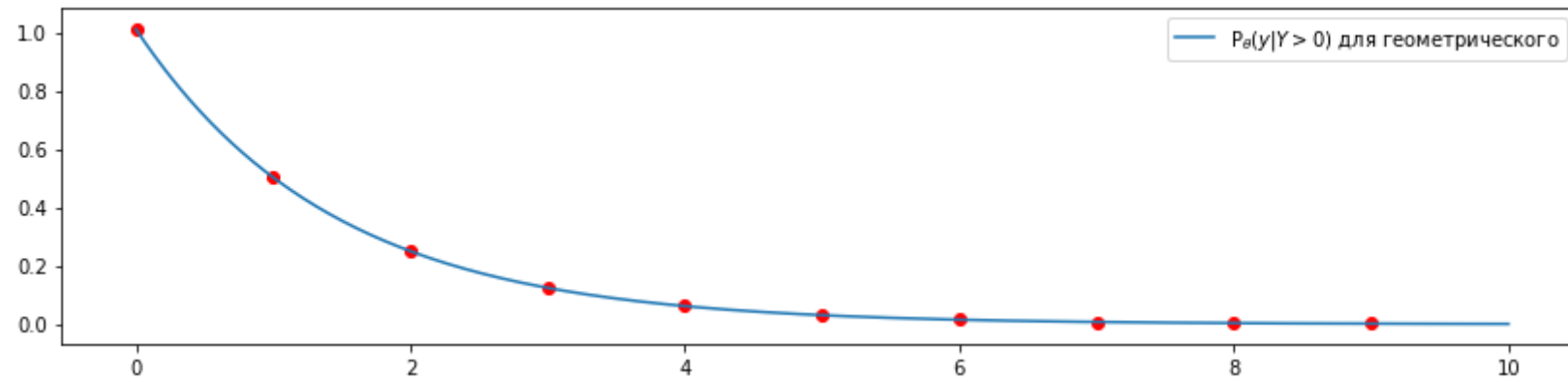
```

In [241]: plt.figure(figsize=(14,7))
from math import gamma
def pois_cond_pmf_gamma(x,mu):
    e = np.exp(- mu)
    return mu ** x * e / (1. - e) / (gamma(x+1))

plt.subplot(211)
grid = np.linspace(0, 10, 1000)
n_grid = np.arange(0, 10, 1)

plt.plot(grid, [geom_cond_pmf(x,p_data) for x in grid], label=
    "$\\mathsf{P}_\\theta$ (y \\left| Y > 0 \\right)$ для геометрического")
plt.scatter(n_grid, [geom_cond_pmf(x,p_data) for x in n_grid], color="red")
plt.legend()
plt.subplot(212)
plt.plot(grid, [pois_cond_pmf_gamma(x,mu_data) for x in grid], label=
    "$\\mathsf{P}_\\theta$ (y \\left| Y > 0 \\right)$ для пуассоновского\n" +
    " (при использовании гамма-функции вместо факториала)")
plt.scatter(n_grid, [pois_cond_pmf(x,mu_data) for x in n_grid], color="red")
plt.legend()
plt.show()

```



Множественное тестирование

```

In [210]: def get_pval(Y, size=100, text=False):
    sub_Y = np.array(np.array(Y)[sps.randint.rvs(0, len(Y), size=size)])

    def to_freq_table(sub_Y, ddof=1):
        Y_table = np.zeros(np.array(sub_Y).max() + 1)
        for y in sub_Y:
            Y_table[y] += 1

        YY = []
        for i in range(ddof, len(Y_table)):
            if (Y_table[i] < 5):
                YY = Y_table[ddof:i]
                YY[-1] += Y_table[i:].sum()
                break
        return YY

    YY = to_freq_table(sub_Y)

    if text:
        print("Количество отцов с (i+1) детьми, с укрупнёнными группами = ", YY)

    p_data = 1. / sub_Y.mean()

    def f(x):
        return x / (1. - np.exp( - x)) - sub_Y.mean()

    mu_data = scipy.optimize.root(f,2.).x

    geom = []
    pois = []

    def geom_cond_pmf(x,p):
        q = 1. - p
        return q ** (x - 1) * p

    def pois_cond_pmf(x,mu):
        e = np.exp(- mu)
        return mu ** x * e / (1. - e) / (factorial(x))

```

```

for i in range(1, len(YY)):
    geom.append(geom_cond_pmf(x=i, p=p_data))
    pois.append(pois_cond_pmf(x=i, mu=mu_data))

geom.append(1. - np.array(geom).sum())
pois.append(1. - np.array(pois).sum())
geom = np.array(geom) * YY.sum()
pois = np.array(pois) * YY.sum()

if text:
    print("YY (данные выборки для теста) = ", YY)
    print("geom (теоретическая выборка из геометрического) = ", geom)
    print("pois (теоретическая выборка из пуассоновского) = ", pois)

pval_geom = sps.chisquare(YY, geom).pvalue
pval_pois = sps.chisquare(YY, pois).pvalue
return [pval_geom, pval_pois]

pvals = np.array([get_pval(Y) for i in range(8)])
pvals

```

```

Out[210]: array([[ 2.84304111e-02,  3.10375384e-07],
 [ 1.99606739e-02,  7.29429771e-05],
 [ 1.66491241e-02,  2.11135868e-07],
 [ 3.30210047e-01,  9.07979124e-06],
 [ 2.34784670e-01,  8.16533628e-06],
 [ 5.31643509e-02,  2.79572787e-05],
 [ 1.02078554e-01,  2.19112658e-04],
 [ 8.51280267e-02,  4.34160864e-06]])

```

```

In [211]: pvals_geom = pvals[:, 0]
pvals_pois = pvals[:, 1]

```

```

In [212]: from statsmodels.sandbox.stats.multicomp import multipletests
multipletests(pvals_geom, alpha=0.05, method='bonferroni')

```

```

Out[212]: (array([False, False, False, False, False, False, False, False], dtype=bool),
 array([ 0.22744329,  0.15968539,  0.13319299,  1.          ,  1.          ,
         0.42531481,  0.81662844,  0.68102421]),
 0.0063911509545450107,
 0.00625)

```

```
In [213]: multipletests(pvals_pois, alpha=0.05, method='bonferroni')
```

```
Out[213]: (array([ True,  True,  True,  True,  True,  True,  True,  True], dtype=bool),
          array([ 2.48300307e-06,  5.83543817e-04,  1.68908694e-06,
                  7.26383299e-05,  6.53226902e-05,  2.23658229e-04,
                  1.75290127e-03,  3.47328691e-05]),
          0.0063911509545450107,
          0.00625)
```

Вывод: множественное тестирование по варианту, предложенному на семинаре, демонстрирует аналогичный ответ:

- в первом тесте нулевая гипотеза, соответствующая геометрическому распределению **не** отвергается на данном α (все тесты на различных подвыборках не отвергают гипотезу)
- во втором тесте нулевая гипотеза, соответствующая пуассоновскому распределению отвергается на данном α (единогласно отвергнута всеми тестами)

Замечание: число гипотез для множественного тестирования выбрано 8, т.к. метод Бонферрони не рекомендуется использовать при большем числе гипотез из-за потери мощности (сложно отвергнуть неверные гипотезы).

```
In [ ]:
```